A cognitive robot equipped with autonomous tool innovation expertise

Handy Wicaksono[1], Claude Sammut[2]
[1]Department of Electrical Engineering, Petra Christian University, Indonesia
[1,2]School of Computer Science and Engineering, University of New South Wales, Australia

Article Info

Article history:

Received Jun 21, 2019
Revised Dec 2, 2019
Accepted Dec 10, 2019

Keywords:

Inductive Logic Programming
Robot learning
Robot tool use
Tool creation
Tool use learning

ABSTRACT

Like a human, a robot may benefit from being able to use a tool to solve a complex task. When an appropriate tool is not available, a very useful ability for a robot is to create a novel one based on its experience. With the advent of inexpensive 3D printing, it is now possible to give robots such an ability, at least to create simple tools. We proposed a method for learning how to use an object as a tool and, if needed, to design and construct a new tool. The robot began by learning an action model of tool use for a PDDL planner by observing a trainer. It then refined the model by learning by trial and error. Tool creation consisted of generalising an existing tool model and generating a novel tool by instantiating the general model. Further learning by experimentation was performed. Reducing the search space of potentially useful tools could be achieved by providing a tool ontology. We then used a constraint solver to obtain numerical parameters from abstract descriptions and use them for a ready-to-print design. We evaluated our system using a simulated and a real Baxter robot in two cases: hook and wedge. We found that our system performs tool creation successfully.

Corresponding Author:

Handy Wicaksono,
Department of Electrical Engineering,
Petra Christian University,
Jl. Siwalankerto 121-31, Siwalankerto, Wonocolo, Surabaya, Jawa Timur, 60236-Indonesia.
Email: handy@petra.ac.id

1. INTRODUCTION

Humans use tools to solves complex problems in their daily life. When an appropriate tool is not available, humans can innovate and design a new tool, often based on prior experience, or even invent it from scratch. In previous work, Brown & Sammut [1] showed how a robot can learn to use an existing object as a tool. However, tool creation is a much more difficult task. The robot must learn the properties of an object that allows it to be used as a tool for a specific task, e.g. a hook-shaped object is good for pulling things out of narrow spaces. If such an object is present in the robot's world, then a planner can incorporate a tool use action into a plan. Where the robot has the ability to create objects of arbitrary shape, for example using a 3D printer, the space of possible objects available has tools is almost infinite. Thus, tool creation is challenging in giving the learning system the ability to innovate, but its search space is potentially huge, and therefore must be constrained. Although there is increasing interest in tool use learning by a robot, there has been little work on tool creation. Wang et al. [2] developed a robot that is able to create tools on the fly, but the tools design are predefined and no learning is performed. One limitation, that we will discuss further, is that most existing work uses feature-based representations [3] are not expressive enough to be extended to tool creation.

Creating a novel tool by modifying or extending a previous design is called **tool innovation** in cognitive science [4]. We take this approach as a means of limiting the search space of possible tools that the robot may try to design. The system uses a (more or less) specific-to-general search, starting with a design of a known tool, but which either is not present or whose properties only partially match the requirements of the task. These properties are then modified by generalisation of the tool description until an appropriate set of properties is obtained.

We introduce a system called CREATIVE (**Cognitive Robot Equipped with Autonomous Tool Innovation Expertise**), which is able to learn how to use a tool and, if needed, design and build a new one. We use a relational representation of tool models that are learned by a form of Inductive Logic Programming (ILP) [5, 6]. A PDDL action model for tool use is initially constructed by observing a trainer. The model is then generalised by trial and error, conducting experiments with different objects of similar, but not identical shape and placing the tools in different poses to learn the structural and spatial properties of an object that make it a suitable as a tool for a particular kind of task.

We propose a novel tool creation mechanism. It proceeds as above except is not limited to only experimenting with existing objects. Instead, by performing further generalisations using the replacement operation of Marvin [7], it can create a completely new object. A constraint solver can then be used to provide bounds on the numerical parameters required to manufacture the object by 3D printing. Unlike Brown & Sammut [1] who performed experiments in a sensor-less simulation, we conduct our experiments using a real Baxter robot, and for testing on large numbers of example, we use a sensor-based simulation of the Baxter. In the following sections, we review related work in tool creation by a robot, describe our representation of states and action, and explain our learning mechanism. Experimental results are then presented and analysed.

2. RELATED WORK

Previous work on tool use learning by a robot includes Wood et al. [8], who used a neural network to learn the posture of a Sony Aibo so that it could reach an object using a tool attached to its back. Stoytchev [9] demonstrated learning tool affordances that are grounded in the robot's behaviours, to pull a hockey ball using a hook-like object. Tikhanoff et al. [10] integrate exploratory behaviours and geometrical feature extraction to learn affordances and tool use. Mar et al. [3] extend their work by learning the grasp configurations that affect the outcome of a tool use action. All of these systems use feature vector representations to describe an object's attributes, and therefore, are limited in their ability to describe relations between components of the tool and relations to other objects. Relational learning methods overcome those limitations [1].

Relational tool use learning is a special case of action learning. EXPO [11] continuously monitors plan execution. When a fault occurs, it performs experiments to find missing preconditions or effects in a planning operator. OBSERVER [12] extends this by learning from a trainer’s solution traces as well. However, EXPO and OBSERVER are not able to deal with noise. TRAIL [13] was the first to attempt at learning action models in a noisy world using ILP. Brown & Sammut [1] applied Explanation-based Learning (EBL) and ILP to learn an action model of a simulated robot that uses tools to achieve its goals.

There has been little prior work on tool creation. Wang et. al. [2] added a thermoplastic adhesive supplier and a thermal connector on a robot arm so that it could manufacture tools on the fly. However, the tool design is predefined, not learned. Hornby et al. [14] developed a generative representation to automatically design the morphology and controller of a simple robot as a component to build a more complex robot. Brodbeck et al. [15] performed evolutionary stochastic optimisation of mechanical designs to construct a complex agent. A “mother” arm robot is assigned to design and assemble the candidate agents. This evolutionary approach is limited to feature-based representations and learning takes a long time as it cannot incorporate any background knowledge nor accumulate learned knowledge. ILP is capable of learning expressive relational representations, it can easily make use of background knowledge and learned models can be added as background knowledge for further learning. We have published a paper on our early idea in tool creation by a robot [16].

3. RESEARCH METHOD

In this section, we will describe our state and action representation, tool use learning and tool creation mechanism.

3.1. State and action

We maintain two levels of state representation: abstract and primitive. Primitive states contain quantitative values, such as the pose of objects in the world, while abstract states capture qualitative relationships between objects. As we operate in an ILP framework, state representations are expressed as a set of Prolog facts. We define a tool as an object that is deliberately employed by an agent to help it achieve a goal that would otherwise be difficult or impossible to achieve. A tool possesses spatial and structural properties. We build a simple hierarchy for a tool's structural properties, distinguishing tools by their structure and size. Our background knowledge for a hook-like tool is shown in Figure 1. We implement it in Prolog.

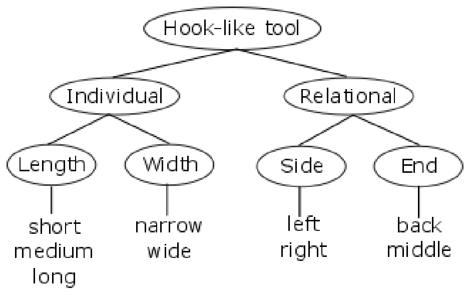

Figure 1. Hierarchy of a hook-like tool's structural properties

We use a STRIPS-like action model [17] to describe tool actions:
- PRE: condition that must hold so that the action can be performed
- EFFECTS: conditions that become true or false as a result of performing the action

The action model does not provide any quantitative information needed to execute a motor command. For example, the action model might state that the hook must be placed behind the object to be pulled but it doesn't not give a precise position. However, spatial literals in the action model's effects (e.g. behind (X, Y)) can be used as constraints by a constraint solver [18] to produce a set of quantitative parameter values, any of which can be used to achieve the effects.

3.2. Tool use learning

CREATIVE consists of tool-use learning and tool creation. Initially, the robot does not have complete action models, so it cannot generate a plan to complete a tool-use task. The robot is shown a correct example by a trainer, which is used to construct the initial action model. There are several stages in this process: acquire and track objects by a vision sensor; segment observations into states and identify potential actions by matching observations with existing action models; and construct a new action model if there are segments that have no match. The precondition of a new action model contains a hypothesis about the tool's required properties.

Our hypothesis formation adapts Mitchell's version space [19] in that the initial example represents the system's most specific hypothesis (h_s) for the tool's structure and pose and the most general hypothesis (h_g), initially, covers everything. The system refines its hypothesis by conducting experiments. It may generalise the most specific hypothesis, then construct an instance that is consistent with its generalisation, and vice versa. While learning to use an existing tool, construction of an instance of an action model consists of selecting an object whose shape is consistent with the structural properties of the hypothesis and positioning it so that it is consistent with the spatial properties of the hypothesis.

If no object is a perfect match, the object that matches the most structural properties of the tool in the trainer's example is chosen. More specifically, we test whether an object satisfies most structural properties in h_g and h_s. To select the pose, firstly, we collect the spatial literals in h_g then append them to the spatial predicates in h_s. Those predicates are then used to construct constraints for a constraint solver. The aim here is to convert logical constraints, into numerical values that can be given to an inverse kinematics solver to place the tool in the desired position. The learning algorithm borrows from Golem [20], which performs a Relative Least General Generalisation (RLGG) [21]. h_s is refined by finding the constrained Least General Generalisation (LGG) of a pair of positive examples, and h_g is refined by performing a mechanism inspired by Negative-based Reduction (NBR). The tool use learning algorithm, extends Brown & Sammut [1] and Haber [22], is shown in tool use learning algorithm in Figure 2.

3.3. Tool creation

Tool creation is triggered if existing tools cannot be used to perform a task. To create a new tool, a robot must have previously learned models of tools that have at least some of the properties required for the new task. The robot is provided with background knowledge to describe the structural properties of simple tools see Figure 1, for hook-like tools. Note that in the present work, this background knowledge is hand crafted, but it may also be learned. The main function of the background knowledge is to limit the search space when generalisation is conducted and is used in much the same way as Shapiro's “refinement graph” in MIS [23].

To learn a tool's functional specification based on knowledge of previous tool use, an agent must generalise the action models that were acquired during the tool-use learning phase. In tool creation, generalisation is done by performing a replacement operation (as in Marvin [7]) with respect to the tool ontology. A new instance, which differs from the previous one, can then be generated and tested. For example, if the current model requires a hook to be on the right hand side of the tool, a generalisation may be that the hook can be on either side.

Having the abstract structural properties of a new tool, its numerical parameters can be acquired by treating the structural properties as constraints for a constraint solver. A tool is then ”manufactured” by converting the tool's parameters into an Open SCAD format, which later can be printed by a 3D printer. In simulation with Gazebo, an SDF (Simulation Description Format) or a URDF (Universal Robotic Description Format) file can be constructed.

Once the candidate tools are realised, further learning by experimentation can be performed to ”debug” the action model. This is again inspired by MIS [23], which debugs logic programs by querying a human oracle. In our case, the oracle is the physics simulator. In the ILP framework, learned action models can be stored as background knowledge for future learning. The aim in tool creation is to generalise the structural properties of a tool and then create new specialisations, suited to a task that is different from previously encountered ones. Our complete tool creation mechanism is described in tool creation algorithm shown in Figure 3. Tool generalisation is performed in steps 1 and 2, while tool manufacturing is performed in steps 3 until 9. Finally, learning by experimentation is done in step 10 to 18. We will elaborate each step in the following sections.

```
Algorithm 1 Tool use learning
1: Input: new action model M, N trials,
      K consecutive success, $h_g$ = true,
      $h_s$ = preconditions of M
2: while $cons\_success < K$ or $index < N$ trials do
3:    $e = generate\_experiment(h_s, h_g)$
4:    $tool = select\_tool(h_s, h_g)$
5:    for each $e_i$ in $e$ do
6:       $pose = select\_pose(e_i)$
7:       if pose null then
8:          break
9:    $success = execute\_experiment(tool, pose)$
10:   if $success$ then
11:      label pose positive
12:      increment $cons\_success$
13:      $h_s$ = generalise $h_s$
14:   else
15:      label pose negative
16:      $cons\_success \leftarrow 0$
17:      $h_g$ = specialise $h_g$
18:   Add $e_i$ to training data
19:   increment trial
```

Figure 2. Tool use learning algorithm

```
Algorithm 2 Tool Creation
Require: current positive example $e^+$, most specific hypothesis $h_s$, back-
   ground_knowledge $B$
1: Choose one property in $e^+$, find relevant replacement operators in $B$, and generalise
   that property
2: Based on previous step, construct a new object or tool to be tested
3: Perform constraint solving on the structural properties of the novel tool to get its
   numerical sizes
4: if Experiments are in the simulation then
5:    Generate an SDF or a URDF file based on the tool's parameters
6:    Spawning the new tool into the simulation environment
7: else if Experiments are in the real world then
8:    Generate an OpenSCAD file based on the tool's parameters
9:    Convert it into STL format and print the new tool using a 3D printer
10: Evaluate the new tool in a tool-use learning experiment
11: if The tool use experiment is successful then
12:    Update the tool_pose hypothesis by performing $LGG$ on the current hypothesis
   and the positive example
13: else
14:    Try to generate another object and test it
15:    if It is successful then
16:       Create a new category to accommodate several successful tools
17:    else if It fails then
18:       Continue to find a new object until there is no literal left in $h_s$
Ensure: updated tool_pose hypothesis $h'_s$
```

Figure 3. Tool creation algorithm

4. RESULTS AND DISCUSSIONS

We evaluate our method on two cases: using a hook to pull a cube out of a tube and using a wedge to prevent a door from closing. We use a Baxter robot, which has a writs-mounted camera, and add two external web cameras in different positions: downward-view and side-view. We only use downward-view camera for the hook example, and but both external cameras are used for the wedge. The complete experimental scenes for those two cases are shown in Figure 4. Our previous work described the robot architecture and its practical implementation in detail [24].

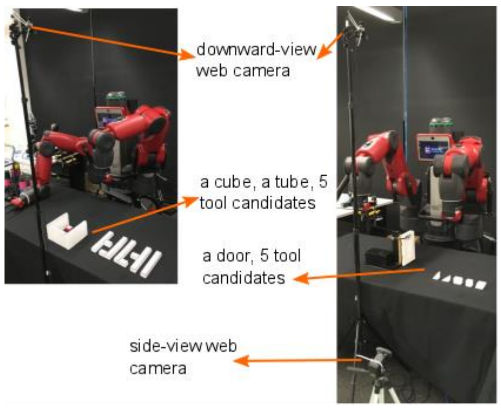

Figure 4. Experimental scene for case 1 (left) and 2 (right)

4.1. Case 1: using a hook to pull a cube out of a tube

In the first case, there are two sets of five hook-like objects where the second set has narrower hooks than the first. A cube is positioned in one of four predefined locations. When a robot is successful in pulling a cube, a new task with a new cube location is given. Otherwise, the robot tries to repeat solving the current task, to conform the result. In the learning experiments, first, a tutor gives an example of pulling out the cube. The learning by observation phase records snapshots of the scene, and from those, constructs qualitative states from primitive ones, matching those states with existing action models, building new action models to accommodate unmatched states.

The above process enables the robot to build complete abstract actions to perform a tool-use task. However, the action models must be refined by trial-and-error learning. ILP, more specifically RLGG, is used for the learning mechanism. To minimise real-world experiments, we perform learning in simulation first. Twelve learning episodes (see Tool Use Learning/TUL trial 1-12 in Figure 5) are required for the robot to learn the model for a reliable tool. After successful simulation, the tool is printed by a 3D printer and tested in the real world with the same mechanism (see TUL trial 13-16). Some results have previously been reported [25].

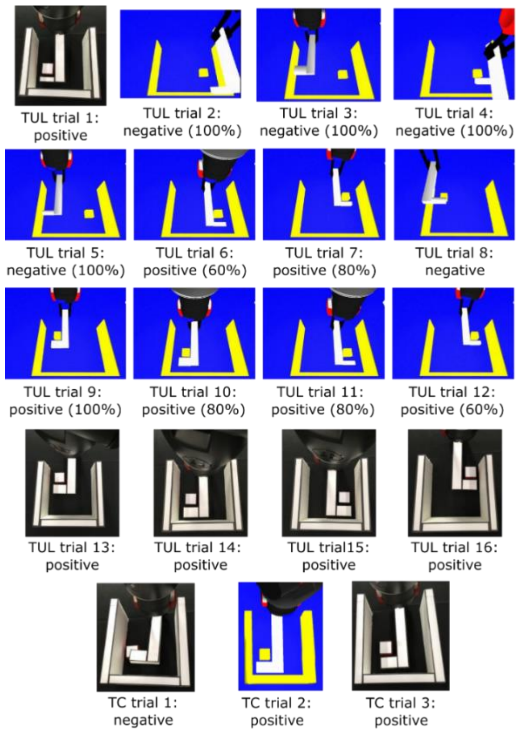
TUL trial 1: positive
TUL trial 2: negative (100%)
TUL trial 3: negative (100%)
TUL trial 4: negative (100%)
TUL trial 5: negative (100%)
TUL trial 6: positive (60%)
TUL trial 7: positive (80%)
TUL trial 8: negative
TUL trial 9: positive (100%)
TUL trial 10: positive (80%)
TUL trial 11: positive (80%)
TUL trial 12: positive (60%)
TUL trial 13: positive
TUL trial 14: positive
TUL trial15: positive
TUL trial 16: positive
TC trial 1: negative
TC trial 2: positive
TC trial 3: positive

Figure 5. Experiments in learning to use and create a hook

We then modify the task by using a longer tube. In this case, the existing tool fails to pull the cube because it is too short (see TC trial 1). We use a replacement operation [7] with respect to the background knowledge:

```
% Background knowledge
any_length(X) :- handle_length(X,medium).
any_length(X) :- handle_length(X,long).
any_length(X) :- handle_length(X,short).
```

There is a literal in h_s(State): "handle_length(X,medium)" which means that the length of a handle of a hook is medium. This can be replaced by "any_length(X)". A new instance is created such that it does not satisfy the replaced literal: "handle_length(X,long)". A constraint solver is used to convert those structural properties into numerical parameters (width and length) of a tool. We perform another tool-use-learning experiment and find that the new tool is able to pull the cube out of the longer tube successfully (TC trial 2: 100\% success rate in 5 experiments). To validate the learning result in the simulated experiment, another experiment in the real world is conducted (see TC trial 3).

4.2. Case 2: using a wedge to prevent a door from closing

In the second case, we want to find an object which is useful as a doorstop. This case is simpler than the first one, as the chosen objects are simple ones (not a composite object), and the task does not have any variations. We provide two sets of objects that have different shapes and different widths. As before, a tutor will give example of correct use of a wedge to stop the door. Initial action models are developed and further trial-and-error learning are performed to get more accurate models. The robot needs 9 trials until it finds the correct shape and pose of an object, see TUL trial 1 until 9 in Figure 6. After learning, the system has found that, to act as a doorstop, an object must have these properties: it has a triangular prism shape, its ratio of length to width (length/width) must be more than one (mechanical advantage of a prism is more than one), it is located near the door, and it is located on the upper side (either middle-upper or top side) of the door.

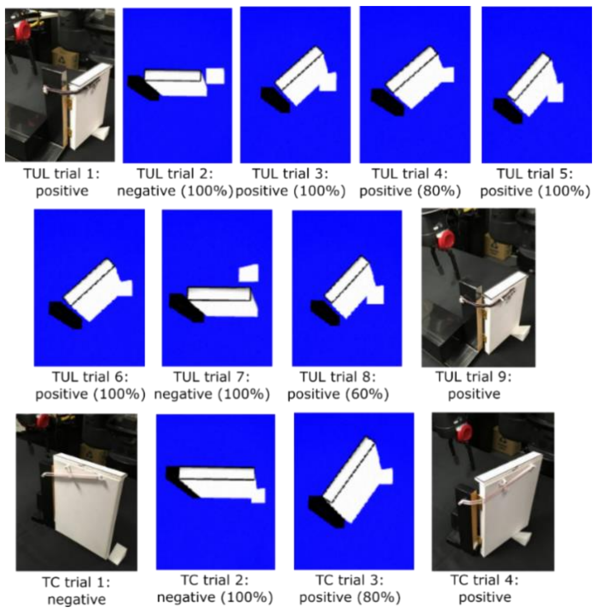
TUL trial 1: positive
TUL trial 2: negative (100%)
TUL trial 3: positive (100%)
TUL trial 4: positive (80%)
TUL trial 5: positive (100%)
TUL trial 6: positive (100%)
TUL trial 7: negative (100%)
TUL trial 8: positive (60%)
TUL trial 9: positive
TC trial 1: negative
TC trial 2: negative (100%)
TC trial 3: positive (80%)
TC trial 4: positive

Figure 6. Experiments in learning to use and create a wedge

We then modify the task by using a bigger door. The existing wedge fails to stop this door, as now its mechanical advantage is not great enough to hold the door. Our robot then performs the replacement operation w.r.t background knowledge. The existing structural property length_s(State,medium) can be generalised to any_length_s(State) and then instantiated with a different value. In TC trial 2, length_s(State,short) is chosen. This new tool is created and tested in simulation. It fails because its mechanical advantage is even smaller than the original wedge. The next trial generalises the same property and chooses the new instance: length_s(State, long). This time it is successful in stopping the big door, because its mechanical advantage is greater than the existing wedge. The new wedge is realised by 3D printing, and another learning experiment is performed. The result is positive, so the robot stops this learning episode.

5. CONCLUSION

We propose a novel mechanism for a robot to learn how to use a tool and create a new one if it is needed. We use a relational representation that is general and flexible and can take advantage of background knowledge for tool creation. A novel tool creation method extends previous tool use learning (based on RLGG) with a replacement operation. We also propose an automatic tool manufacturing method by converting abstract structural properties into numerical parameters using a constraint solver. Those parameters are used to manufacture a new design that later can be realised by 3D printing. We evaluate our approach on two tasks: using a hook to pull a cube out of a tube, and using a wedge to prevent the door from closing. Tool use learning and tool creation can be done successfully in those cases.

ACKNOWLEDGEMENTS

Handy Wicaksono is supported by DG-RSTHE, Ministry of Research, Technology, and Higher Education of the Republic of Indonesia.

REFERENCES

[1] S. Brown and C. Sammut, "A relational approach to tool-use learning in robots," in *International Conference on Inductive Logic Programming*, pp. 1-15, 2012.
[2] L. Wang, L. Brodbeck, and I. Fumiya, "Mechanics and energetics in tool manufacture and use: a synthetic approach," *Journal of the Royal Society Interface*, vol. 11, no. 100, 2014.
[3] T. Mar, *et al.*, "Self-supervised learning of grasp dependent tool affordances on the iCub Humanoid robot," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015.
[4] S. R. Beck, *et al.*, "Making tools isn't child's play," *Cognition*, vol. 119.2, pp. 301-306, 2011.
[5] C. Sammut, "Concept Learning by Experiment," in *Proceedings of IJCAI*, pp. 104-105, 1981.
[6] S. Muggleton, "Inductive logic programming," *New generation computing*, vol. 8, no. 4, pp. 295-318, 1991.
[7] C. Sammut, and R. B. Banerji, "Learning concepts by asking questions," *Machine learning: An artificial intelligence approach*, vol. 2, pp. 167-192, 1986.
[8] A. B. Wood, *et al*., "Effective tool use in a habile agent," In *2005 IEEE Design Symposium, Systems and Information Engineering*, pp. 75-81, IEEE, 2005.
[9] A. Stoytchev, "Behavior-grounded representation of tool affordances," In *Proceedings of the IEEE international conference on robotics and automation*, pp. 3060-3065, IEEE, 2005.
[10] V. Tikhanoff, *et al*., "Exploring affordances and tool use on the iCub," in *13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 130-137, IEEE, 2013.
[11] Y. Gil, "Learning by experimentation: Incremental refinement of incomplete planning domains," in *Machine Learning Proceedings 1994*, pp. 87-95, 1994.
[12] X. Wang, "Learning by observation and practice: An incremental approach for planning operator acquisition," In *Machine Learning Proceedings 1995*, pp. 549-557, 1995.
[13] S. Benson, "Inductive learning of reactive action models," in *Machine Learning Proceedings*, pp. 47-54, 1995.
[14] G. S. Hornby, *et al.* "Generative representations for the automated design of modular physical robots," *IEEE transactions on Robotics and Automation,* vol. 19, no. 4, pp. 703-719, 2003.
[15] L. Brodbeck, *et al*., "Morphological evolution of physical robots through model-free phenotype development," *PloS one*, vol. 10, no. 6, 2015.
[16] C. Sammut, R. Sheh, A. Haber and H Wicaksono, "The Robot Engineer," in *ILP (Late Breaking Papers)*, pp. 101-106, 2015.
[17] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial intelligence,* vol. 2, no. 3-4, pp. 189-208, 1971.
[18] K. R. Apt and M. Wallace, *Constraint logic programming using ECLiPSe,* Cambridge University Press, 2006.
[19] T. M. Mitchell, "Version spaces: A candidate elimination approach to rule learning," in *Proceedings of the 5th international joint conference on Artificial intelligence*, vol. *1*, pp. 305-310, 1977.
[20] S. Muggleton and C. Feng, "Efficient induction of logic programs," *New generation computing*, pp. 368-381, 1990.
[21] G. D. Plotkin, "A note on inductive generalization," *Machine intelligence*, vol. 5, no. 1, pp. 153-163, 1970.
[22] A. Haber, "A system architecture for learning robots," *PhD Thesis*, The University of New South Wales, 2015.
[23] E. Y. Shapiro, "Algorithmic program debugging," *PhD Thesis*, 1982.
[24] H. Wicaksono and C. Sammut, "Tool Use Learning for a Real Robot," *International Journal of Electrical and Computer Engineering*, vol. 8, no. 2, 2018.
[25] H. Wicaksono and C. Sammut, "Relational tool use learning by a robot in a real and simulated world," in *Proceedings of ACRA*, 2016.

BIOGRAPHIES OF AUTHORS

Handy Wicaksono is a Ph.D. student at Artificial Intelligence Group, School of Computer Science and Engineering, UNSW Australia, with bachelor and master degree in Electrical Engineering from Institut Teknologi Sepuluh Nopember, Indonesia. He is also a lecturer in Electrical Engineering Department, Petra Christian University, Indonesia. His research is in the area of artificial intelligence, intelligent robot, and industrial automation.

Claude Sammut is a Professor in the School of Computer Science and Engineering at the University of New South Wales. He is Head of the Artificial Intelligence Research Group and Deputy Director of the iCinema Centre for Interactive Cinema Research. His early work on relational learning helped to the lay the foundations for the field of *Inductive Logic Programming* (ILP). With Donald Michie, he also did pioneering work in *Behavioural Cloning.* His current interests include *Conversational Agents* and *Robotics*. He was a member of the executive committee of the RoboCup Federation from 2003 to 2009 and in 2012 was elected to the board of trustees of the RoboCup Federation and is co-editor-in-chief of Springer's Encyclopedia of Machine Learning and Data Mining. He is the general chair of RoboCup 2019, to be held in Sydney.